择排序、插入排序、归并排序、快速排序和堆排序等。如果排序方法不是基于比较的，则可以简称为非比较排序。常见的非比较排序方法包括基数排序和计数排序等。桶排序是允许多种排序策略并存的混杂方法，但是其主要框架是非比较排序，因此常被分类为非比较排序方法。

在评估排序算法的性能时，一般需要考虑如下因素。

(1) 时间复杂度(time complexity)：即算法对一个序列进行排序时需要运算的次数关于序列长度的数量级。可以证明，基于比较的排序算法的平均时间复杂度不可能低于 $O(n\log n)$，而部分非比较排序算法则可以突破这个复杂度瓶颈。

(2) 空间复杂度(space complexity)：即算法在排序过程中需要使用的内存空间关于序列长度的数量级。在排序算法的评估中，一般不考虑存储序列本身所用的空间，而是评估排序算法所用的额外空间的复杂度。

(3) 稳定性(stability)：对于稳定的排序算法，序列中若存在若干相同的元素，排序后相同元素的相对位置不变。插入排序、冒泡排序、归并排序、计数排序等算法通常是稳定的，选择排序、快速排序、堆排序等算法通常是不稳定的。具体的排序算法是否稳定与代码实现有关。

常见的比较排序方法如表 1.11 所示(n 为待排序元素的数量，稳定性仅就排序方法的常见实现而言，下同)。

表 1.11　常见的比较排序方法

方　法	平均时间复杂度	最坏时间复杂度	空间复杂度	是否稳定	备注
冒泡排序	$O(n^2)$	$O(n^2)$	$O(1)$	✓	
选择排序	$O(n^2)$	$O(n^2)$	$O(1)$	✗	如果使用 $O(n)$ 的额外空间，可以做到稳定排序，此时需要使用链表，或采用元素插入而非交换方式
插入排序	$O(n^2)$	$O(n^2)$	$O(1)$	✓	最坏时间复杂度可以更准确地表示为 $O(n+d)$，其中 d 为原始序列中的逆序数量
归并排序	$O(n\log n)$	$O(n\log n)$	$O(n)$	✓	
快速排序	$O(n\log n)$	$O(n^2)$	$O(\log n)$	✗	可用原址(in-place)操作来减少存储空间占用
堆排序	$O(n\log n)$	$O(n\log n)$	$O(1)$	✗	

常见的非比较排序方法如表 1.12 所示。

表 1.12　常见的非比较排序方法

方法	平均时间复杂度	最坏时间复杂度	空间复杂度	是否稳定	备注
基数排序	$O\left(kn+\sum_{i=1}^{k}w_i\right)$	$O\left(kn+\sum_{i=1}^{k}w_i\right)$	$O(n+\max_i w_i)$	✓	k 为关键值的划分个数，w_i 为第 i 个关键值划分的值域规模
计数排序	$O(n+w)$	$O(n+w)$	$O(n+w)$	✓	w 为值域规模
桶排序	$O\left(n+\frac{n^2}{m}+m\right)$	$O(n^2)$	$O(n+m)$	✓	对应 m 个桶且桶内做插入排序的情形。如果在桶内采用归并排序等，最坏时间复杂度为 $O(n\log n)$

参考词条

1. 算法概念
2. 时间复杂度分析
3. 空间复杂度分析

延伸阅读

[1] 徐孝凯. 数据结构实用教程[M]. 2版. 北京：清华大学出版社，2006：343-344.

[2] DONALD E K. 计算机程序设计艺术 卷3：排序与查找[M]. 2版. 北京：人民邮电出版社，2017：1-61.

典型题目

NOIP2009 普及组 分数线划定

（赵启阳　谢秋锋）

1.4.5.2 冒泡排序

冒泡排序是一种基于相邻元素比较的简单排序算法。其基本思想是通过相邻元素之间的比较和交换使排序码较大的元素逐渐从顶部移向底部，而排序码较小的元素逐渐上移，就像水底下的气泡一样逐渐上冒。对于长度为 n 的序列 a，冒泡排序算法的主要步骤如下：

（1）扫描序列 $n-1$ 次；

（2）第 i 次扫描，从前往后比较相邻的两个元素，即比较 $a[j]$ 与 $a[j+1]$，其中 $1 \leqslant j \leqslant n-i$；

（3）每次比较时，若靠前的元素比靠后的元素大，则交换它们，否则不做操作。

冒泡排序需要扫描序列 $n-1$ 次，第 i 次扫描进行 $n-i$ 次比较操作，总共比较 $(n-1)+(n-2)+\cdots+1=\dfrac{n(n-1)}{2}$ 次，故时间复杂度为 $O(n^2)$。

冒泡排序算法的额外空间复杂度为 $O(1)$。

冒泡排序算法在遇到两个相邻的相同元素时不会产生交换，故冒泡排序是稳定的算法。

冒泡排序的优化方法：如果某次扫描没有元素进行交换，说明序列已经有序，可以结束冒泡排序。使用该方法，最优情况下初始序列有序，则只需要进行一次扫描，最优时间复杂度为 $O(n)$；最坏情况下初始序列反序，冒泡排序仍需扫描 $n-1$ 次，故最坏时间复杂度为 $O(n^2)$。

代码示例

优化后的冒泡排序算法核心代码如下。

```
//优化后的冒泡排序
for (int i = 1; i <= n - 1; i++)
{                          // 扫描序列 n-1 次
    bool flag = false; // 记录本次扫描是否交换了元素
    for (int j = 1; j <= n - i; j++)
    { // 从前往后比较相邻的两个元素
        if (a[j] > a[j + 1])
        {                              // 如果前面的元素大于后面的元素
            swap(a[j],a[j + 1]); // 交换它们
            flag = true;
        }
    }
    if (flag == false)
        break; // 如果没有产生交换则排列有序,跳出循环
}
```

参考词条

排序的基本概念

延伸阅读

[1] 徐孝凯. 数据结构实用教程[M]. 2 版. 北京：清华大学出版社，2006：352-354.

[2] DONALD E K. 计算机程序设计艺术 卷 3：排序与查找[M]. 2 版. 北京：人民邮电出版社，2017：81-87.

（谢秋锋）

1.4.5.3 选择排序

选择排序是从待排序的区间中选择出具有最小排序码的元素，并将该元素与该区间的第 1 个元素交换位置。这样通过 $n-1$ 次选择未排序部分的最小元素，将其放在正确位置从而达到对整个序列进行排序的效果。对于长度为 n 的序列，选择排序算法的主要步骤如下。

(1) 扫描序列 $n-1$ 次。

(2) 第 $i(1\leqslant i\leqslant n-1)$ 次扫描，找到待排序区间 $a[i],a[i+1],\cdots,a[n]$ 中的最小元素 $a[m]$。

(3) 交换 $a[i]$ 与 $a[m]$。这样 $a[1],a[2],\cdots,a[i]$ 储存的恰好依次是序列中的第 $1,2,\cdots,i$ 项元素。

选择排序进行 $n-1$ 次扫描，第 i 次扫描 $n-i+1$ 个元素，扫描结束进行一次交换，总共扫描 $n+(n-1)+\cdots+2=\frac{n(n+1)}{2}-1$ 个元素，故时间复杂度为 $O(n^2)$。

选择排序本身只需要维护一个 m 变量存储每次扫描的最小元素的位置，故额外空间复杂度为 $O(1)$。

与冒泡排序不同，选择排序并不是稳定的排序，因为其涉及不相邻位置的元素的交换。如序列(2,2,1)，在第一次扫描后 1 与第一个 2 交换，那么在最终序列中，原始序列中的第一个 2 被换到第二个 2 后面了，因此选择排序并不稳定。

代码示例

选择排序的核心代码如下。

```
for (int i = 1; i <= n - 1; i++)
{ // 扫描序列 n-1 次
    int pos = i;
    for (int j = i + 1; j <= n; j++)
    {
        if (a[j] < a[pos])
        {
            pos = j; // 寻找 a[i] 到 a[n] 中的最小元素的位置
        }
    }
    swap(a[i],a[pos]); // 将最小元素放在第 i 个位置
}
```

参考词条

排序的基本概念

延伸阅读

[1] 徐孝凯．数据结构实用教程[M]．2 版．北京：清华大学出版社，2006：347-348.

[2] DONALD E K. 计算机程序设计艺术 卷 3：排序与查找[M]．2 版．北京：人民邮电出版社，2017：107-112.

（谢秋锋）

1.4.5.4 插入排序

插入排序是一种将待排序元素依次加入到已排序序列中的恰当位置，最终形成有序序列的方法。对于长度为 n 的序列，插入排序的主要步骤如下：

(1) 从第 2 个元素开始，执行步骤 $n-1$ 次；

(2) 第 i 次步骤，需要将第 i+1 个元素插入到已经排好序的前 i 个元素中。

在实现时，一般先将第 i+1 个元素存储到临时变量 t 中，从第 i 个元素开始，倒序将每个元素的值与 t 的值比较，如果当前元素的值比 t 大，那么将该元素后移一个位置；否则将 t 的值放置到当前元素的后一个位置，并结束比较。

插入排序的时间复杂度为 $O(n^2)$，插入排序是稳定的排序方法。

代码示例

插入排序的实现代码如下。

```
//对数组 a 进行排序
for (int i = 2; i <= n; i++)
{
    int t = a[i];
    int j = i - 1;
    while (a[j] > t && j >= 1)
    {
        a[j + 1] = a[j];
        j--;
    }
    a[j + 1] = t;
}
```

参考词条

排序的基本概念

延伸阅读

[1] 徐孝凯．数据结构实用教程[M]．2 版．北京：清华大学出版社，2006：347-348.

[2] DONALD E K. 计算机程序设计艺术 卷 3：排序与查找[M]．2 版．北京：人民邮电出版社，2017：107-112.

（谢秋锋）

1.4.5.5 计数排序

计数排序是对于元素值域在特定范围内的整数序列的一种排序方法，不是一种基于比较的方法。其基本思路是通过一个大小为值域的数组，统计每个元素的值的出现次数，利用前缀和思路，对每个值求出序列中小于等于该值的元素数量，即可确定序列中每个元素排序后的位置，最终得到排序结果。

计数排序的时间复杂度为 $O(n+w)$，其中 n 为数组大小，w 为值域大小。计数排序是稳定的排序方法。